Software Rejuvenation for Secure Tracking Control

Raffaele Romagnoli[†], Dionisio de Niz[‡], Bruce H. Krogh[‡] and Bruno Sinopoli[†]
[†]Dept.of Electrical and Computer Engineering
[‡]Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA USA
{rromagno|dionisio|krogh|brunos}@andrew.cmu.edu

***Abstract*— Software rejuvenation protects cyber-physical systems (CSPs) against cyber attacks on the run-time code by periodically refreshing the system with an uncorrupted software image. The system is vulnerable to attacks when it is communicating with other agents. Security is assured during the software refresh and re-initialization by turning off all communication. Although software rejuvenation has been demonstrated for some simple systems, many problems need to be addressed to make it viable for real applications. This paper expands the scope of CPS applications for which software rejuvenation can be implemented by introducing architectural and algorithmic features to support trajectory tracking. Following each software refresh, while communication is still off, a safety controller is executed to bring the system into a sufficiently small neighborhood of the current point on the reference trajectory. Communication is then re-established and the reference trajectory tracking controller is resumed, with the potential for being compromised by an attack. A protected, verified hypervisor manages the software rejuvenation sequence and delivers trusted reference trajectory points, which may be received from untrusted communication, but are verified using an authentication process. We present the approach to designing the safety controller and timing parameters and illustrate the performance and design tradeoffs using the PX4 real-time simulator. The concluding section discusses directions for further research.**

I. INTRODUCTION

Cyber security has become a significant research and development issue for control of cyber-physical systems (CPSs), particularly for safety critical applications [1], [2], [3], [4]. The primary goal is to prevent physical damage to the system or environment due to malicious cyber attacks. Standard approaches to security rely on effective attack models. But attacks can be perpetrated in many different ways, and it is impossible to model all possible attacks. Moreover, even the best detection mechanisms are impotent if an attacker is able to modify the run-time software.

Software rejuvenation, an established method for dealing with so-called software aging in traditional computing systems [5], has been proposed recently to deal with unmodeled and undetectable cyber attacks on CPSs [6], [7]. The idea is to periodically refresh the run-time system completely with a trusted, secure copy of the control software to thwart attacks that may have changed the on-line code. Although the basic concept has been implemented for some demonstration systems, timing bounds for the software rejuvenation schedule need to be obtained to assure that the software is refreshed before any damage has been done from malicious software and that the CPS will remain safe and viable after restarting the control software.

The concept of software rejuvenation was introduced by Huang et al. in 1995 [5] to address the problem of so-called software aging; that is, failures that occur when a running program encounters a state that was not anticipated when the software was designed. The basic idea is to restart the software intermittently at a "clean" state, either through complete system reboot or by returning to a recent checkpoint, with the hope that this will prolong the time until unanticipated states occur that might cause failures. Since the introduction of the concept, there has been considerable research into the development and performance of software rejuvenation strategies [8], and it has become a practical tool for enhancing the robustness of many computing systems [9].

Although software aging remains the primary motivation for implementing software rejuvenation strategies in computing systems, a few papers have proposed software rejuvenation to enhance system security [10], [11]. In contrast to software aging where mean-time to failure can be the basis for timing software refresh, the frequency of software refresh to defend against malicious attacks must be determined by the length of time a system can remain viable once its security has been violated.

Arroyo et al. [6], [7] propose software rejuvenation as a strategy for CPS security and demonstrated the concept for a quadrotor controller and an automotive engine controller. These examples illustrate how refreshing software in a CPS impacts performance and introduces timing constraints and safety considerations that aren't present in traditional computing systems.

Abidi et al. [12] develop software rejuvenation for CPS further by introducing three concepts. First, the *hardware root of trust* is a secure onboard module that hosts the capabilities that must be available without compromise to implement software rejuvenation. Second, the *secure execution interval* (SEI) is a period during which all external communication is disabled so that no cyber attacks can occur as the software is refreshed. Third, the *safety controller*, which executes immediately following a software refresh and during the SEI, drives the CPS to a known safe state before restoring communication and returning the system to mission control with vulnerability to attacks. Abidi et al. use

a simple, conservative reachability algorithm to determine the time that can be allowed before the next software reset and illustrate their approach for a simulated warehouse temperature controller and a bench-top 3-DOF helicopter.

In this paper, we adopt the overall approach to software rejuvenation from [12], but modify the timing strategy based on analysis of the control problem. We formulate the software rejuvenation problem for CPS applications where the safety controller is designed using the linearized dynamics to take the system to a neighborhood of a given equilibrium state. Using invariant sets for the safety controller and system operating constraints, we derive bounds on the timing parameters for software rejuvenation strategy. We also introduce the concept of adding control constraints enforced by the hardware root of trust to increase the time allowed for mission control and software reset. We demonstrate our contributions using simulation of the nonlinear dynamics for a 6-DOF quadrotor.

II. TRACKING CONTROL

In CPS applications involving motion are typically controlled hierarchically, as illustrated in Figure. A mission controller defines a sequence of waypoints to be visited. A reference governor generates a sequence of points, called the reference trajectory, to take the system between the waypoints along a safe and feasible trajectory. Motion along the reference trajectory is executed by a tracking controller that generates the commands to the system actuators. Depending on the application, the mission controller and reference governor may be onboard or off-board, but the trajectory controller usually needs to be onboard to satisfy the need for real-time execution of the trajectory. This paper focuses on the implementation of the tracking controller.

The tracking controller problem is often implemented as a regulator that drives the system to a given set point, and motion is accomplished by switching along the points on the reference trajectory as a sequence of set points. Since it is not necessary to actually stop at the points on the reference trajectory, the switching from the current reference point to the next reference point can be implemented using a time-based or event-based scheme. In time-based schemes, the points on the reference trajectory have time stamps and the set point for the tracking controller is updated as time evolves. In event-based schemes, the set point for the tracking controller is updated when the system is within a given neighborhood of the current set point. The secure tracking control proposed in this paper uses an event-based scheme to update the set points for the reference controller.

We assume the system has m continuous control inputs, the n continuous state variables of the system are measurable and the reference points along the reference trajectory are equilibria points for the system. Let $x_{eq} \in \mathbb{R}^n$ denote the current equilibrium reference point with corresponding equilibrium control $u_{eq}\mathbb{R}^m$, and let x denote the deviation of the actual state from x_{eq}. If x is sufficiently small (i.e., if the state of the system is sufficiently close to x_{ref}), the system dynamics in a neighborhood of x_{eq} containing x can be represented by the linear state equation

$$\dot{x} = Ax + Bu + Dd, \tag{1}$$

where u is the deviation of the control input about u_{eq} and d represents any disturbances to the system. In this paper we will assume d is negligible. A stabilizing state feedback control can then be found of the form

$$u = Kx, \tag{2}$$

where $K \in \mathbb{R}^{m\times n}$ and $A + BK$ is stable. If the reference point is not changed, the tracking controller with the state feedback control (2) would drive the system to the reference point, x_{eq}. If x_{eq} is not the final point in the reference trajectory, the reference point can be updated to the next reference point (denoted by, say, x'_{eq}) before x_{eq} is reached. In the following, we introduce the condition for switching to the next reference point as reaching an ellipsoidal neighborhood of the current equilibrium given by

$$\mathcal{E}_{\epsilon'} \triangleq \{x|\ \|x\|_P^2 \leq \epsilon'\}, \tag{3}$$

where ϵ' >0, $P \in \mathbb{R}^{n\times n}$ is a particular positive definite symmetric matrix and

$$\|x\|_P^2 \triangleq x^T P x. \tag{4}$$

III. SOFTWARE REJUVENATION OVERVIEW

Figure 1 illustrates the three operating modes for a CPS with software rejuvenation: *mission control* (MC), where the CPS is executing its intended mission with network communication to other agents in the overall system, including possibly a supervisor; *software refresh* (SR), when a secure copy of the operating software is reloaded to eliminate any possible corruption of the run-time code and data; and *safety control* (SC), which is described below. The period for each of these operating modes is denoted T_{MC}, T_{SR} and T_{SC}, respectively.

Mission control is terminated by the timeout of the *refresh clock*. To assure software refresh and safety control executed without interference from potential attackers, external communication is shutoff and the CPS operates autonomously during the SEI ($T_{SR} + T_{SC}$). After execution of the safety control, the refresh clock is restarted with the refresh clock period t_r (which determines T_{MC}, communication is reestablished and th CPS returns to mission control. Attacks can occur when there is communication. If there is an attack, the control actions may not be known to the CPS controller. This uncertainty about the control actions (and the state of the CPS) remains through the software refresh.

The pseudo code in Alg. 1 illustrates the implementation of the overall software rejuvenation strategy. The protected secure hardware hosts: system communication; trusted software image; refresh clock; and control limits that can be imposed during MC and software SR to constrain what an adversary could do during this period of uncertain control. Since the system is operating securely during SC, these limits can be removed for SC so that it can use the full control capability.

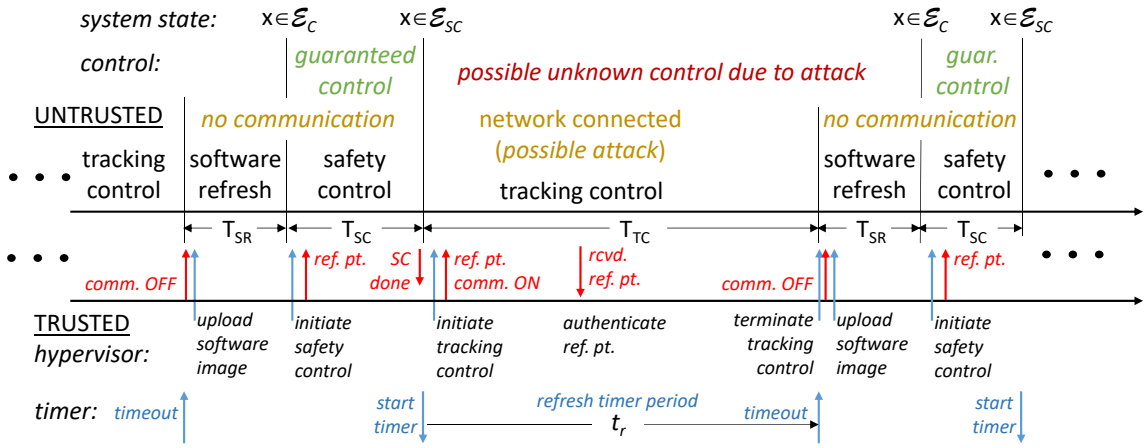

Fig. 1. Software rejuvenation control modes.

Algorithm 1 Software Rejuvenation Algorithm

```
Initialize:
    COMMUNICATION OFF; (protected)
    LOAD SOFTWARE; (protected)
Step 1:
    OFF → CTRL_LIMS; (protected)
    WHILE x ∉ E_ε
        SAFETY_CONTROL;
    END
    ON → CTRL_LIMS; (protected)
    t_r → REFRESH_CLK; (protected)
    COMMUNICATION ON; (protected)
Step 2:
    UNTIL timeout ← REFRESH_CLK (protected)
        MISSION_CONTROL;
    END
Step 3:
    COMMUNICATION OFF; (protected)
    REFRESH SOFTWARE; (protected)
    GO TO Step 1;
```

- UNTRUSTED PARTITION
  - Tracking Control (TC)
  - Safety Controller (SC)
- TRUSTED PARTITION (Hypervisor)
  - Software image
  - Reference trajectory management
    - Authentication
    - Update
    - Send current reference point to SC and TC

1) Secure Reference Trajectory Management: The reference trajectory is a sequence of system equilibrium points delivered by the supervisory controller to the tracking controller to execute the mission. Since this requires communication with the supervisor, the reference trajectory cannot be trusted. Figure X illustrates how the reference trajectory is managed to assure that the CPS is always safe.

To guarantee unknown control actions do not drive the CPS into unsafe or undesired operating states, certain guarantees need to be enforced about the possible states reached during the period of *uncertain control* (UC), $T_{UC} = T_{MC} + T_{SR}$. This is the period when an attacker may have taken over control of the system. The conditions that need to be satisfied are represented by two sets of states, the *safe set* and the *inner safe set*, denoted $\mathcal{E}_\mathcal{C}$ and $\mathcal{E}_\epsilon$, respectively. Figure 2 illustrates the role of these set. The safe set represents general operating constraints to assure system safety and controllability. The goal of the safety controller is to assure the system remains in $\mathcal{E}_\mathcal{C}$ by always returning the state to the inner-safe set before handing control back to the mission controller. During the period T_{UC}, the state may leave $\mathcal{E}_\epsilon$, but safety control must be executed before the state leaves $\mathcal{E}_\mathcal{C}$.

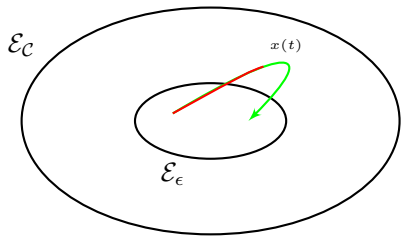

Fig. 2. State constraints for software rejuvenation. Red: uncertain control; green: safety control.

General conditions that guarantee the software rejuvenation algorithm will keep the system in $\mathcal{E}_\mathcal{C}$ are summarized by the following proposition [13].

Proposition 3.1: Given a CPS with time-invariant dynamics and a set of safe states $\mathcal{E}_\mathcal{C}$, a set of inner safe states $\mathcal{E}_\epsilon \subset \mathcal{E}_\mathcal{C}$, software refresh time T_{SR} and a safety controller SC, if

i. $x(0) \in \mathcal{E}_\mathcal{C}$;
ii. $\exists\ \overline{T}_{SC} > 0\ \ni\ \forall\ x \in \mathcal{E}_\mathcal{C},\ \mathcal{R}(x, \overline{T}_{SC}; SC) \subseteq \mathcal{E}_\epsilon$; and

iii. $\exists\ T_{UC} > T_{SR} \ni\ \forall\ 0 \leq t \leq T_{UC},\ \mathcal{R}(\mathcal{E}_\epsilon, t; UC) \subseteq \mathcal{E}_\mathcal{C}$;
then for the CPS controlled by the software rejuvenation algorithm with $t_r = T_{UC} - T_{SR},\ x(t) \in \mathcal{E}_\mathcal{C}\ \forall\ t \geq 0$.

Proposition 3.1 provides conditions for designing safe software rejuvenation strategies. Condition (i) requires that the system starts at a safe state. Condition (ii) indicates the safety controller must be able to drive the system from all safe states to the inner safe set in bounded time. The amount of time SC executes depends on how long it takes to drive the state to $\mathcal{E}_\epsilon$ (Alg. 1 line 6). Condition (iii) indicates that T_{UC} must be sufficient for software refresh and small enough that no admissible control can take the system state from the inner safe set to an unsafe state before safety control is initiated.

The software refresh time T_{SR} is determined by the system hardware and the size of the software image. $\mathcal{E}_\mathcal{C}$ should be made as large as possible to maximize the time allowed for mission control. For a given SC, design of the software rejuvenation algorithm depends on the size of $\mathcal{E}_\epsilon$ and the control constraints imposed during T_{MC} and T_{SR}. T_{UC} can be increased by reducing the size of $\mathcal{E}_\epsilon$ and by making the control limits more conservative (Alg. 1 line 9). But a smaller $\mathcal{E}_\epsilon$ can lead to longer times for SC and tighter control limits reduce the control available for MC, so there are clear design trade offs to be considered.

IV. SECURE TRACKING CONTROL

The achieve secure tracking control, it must be guaranteed that when the tracking controller updates the reference point, the safety controller will be able to drive the system to the ϵ_{SC} neighborhood of the *new* reference point. This leads to the following condition that must be satisfied by the new reference point. Let x_{eq}, x'_{eq} be the current and next reference point, respectively. To guarantee that the SC can drive the system to x'_{eq}, it is necessary that

$$\mathcal{E}_{\epsilon_{TC}}(x_{eq}) \subset \mathcal{E}_{\epsilon_{SC}}(x'_{eq}), \tag{5}$$

This is illustrated in Figure.

V. EXAMPLE: DRONE CONTROL

The proposed software rejuvenation strategy is applied on 6-DOF UAV systems. Specifically a "Generic 10" Quadrotor + geometry" airframe [14] with full thrust of 4 N, and full torque of 0.05 Nm for each motor is considered. The simulations are performed using JMAVSim flight simulator. A 6-DOF quadrotor is a nonlinear system described by a 12 dimensional state space model that contains position, orientation and linear and angular velocities of the UAV. [15]. There are for controls the normal force to the airframe and three torques related to each angle that describes the UAV orientation. Those controls are mapped by a linear application ("mixer") to the space of the control signals of the four motors. That signal represents a percentage of the maximal power that can be generated by each motor.

The UAV is stabilized by an LQR controller designed considering the linearized model of the system around an equilibrium point x_{eq}, that without loss of generality can be considered the origin of the state space [16] [17] [18] .

In the case of the quadrotor, an equilibrium point corresponds to the hovering around a certain altitude. Supposing there are no constraints on the position space, then each point of can be an equilibrium point for this kind of system. Since in this paper the tracking problem is considered switching from an equilibrium point to another, each set refers to the particular equilibrium point x_{eq}. Without loss of generality and accordingly with the theory for continuous-time linear system and invariant ellipsoids reported in Appendix A, the analysis reported in the sequel is performed for the equilibrium point $x_{eq} = 0$.

The LQR controller is computed as in (9) as reported in the Appendix, using $Q = 50 \cdot I_{12}$ and $R = 100 \cdot I_{12}$ where I_{12} is the identity matrix. The matrix K is reported in (22).

The safe set $\mathcal{C}$ is a convex polyhedral subset of the state space $\mathbb{R}^{12}$ which impose the following constraints on

- the position: ± 2.5 m along z (vertical direction), and ± 1 m along x and y (horizontal direction);
- the orientation: the roll ϕ, pitch θ and yaw ψ angles are bounded between $\pm\pi/4$;
- the linear velocities: ± 5 m/s for the vertical and ± 2 m/s for the horizontals directions;
- the angular velocities: ± 5 rad/s for all the angles.

Considering (11), the safe set $\mathcal{E}_\mathcal{C}(x_{eq})$ is computed solving the maximization problem (13) which generates the symmetric $P > 0$ reported in (**??**).

In Fig. 3, $\mathcal{E}_C(x_{eq})$ for $x_{eq} = 0$ is reported in gray

During SC, the controls are bounded by the nominal values, namely 16N for the total normal force to the air frame and ± 0.66 Nm for the torques related to the roll and pitch angles, and ± 0.1 Nm for the yaw. During TC the bounds on the torques are reduced to ± 0.0033 Nm for the former and ± 0.0005 Nm for the latter. Instead the considered total normal force is of 14 N, and the slowest value admissible in case of shut down is 2 N instead of 0 N. The time needed for SR considered in this simulation is 0.1 s.

Using the above set-up and applying the proposed analysis a feasible value of T_{UC} is 0.34 s for $\epsilon = 0.05$, with $\epsilon' = 0.01$.

Figure 7 shows projections of the ellipsoids $\mathcal{E}_\mathcal{C}$ (light blue) and $\mathcal{E}_\epsilon$ (light green) on the space of the positions (left) and angles (right). The figure also shows the projections of the reach set $\mathcal{R}^+(\mathcal{P}_\epsilon, T_{UC})$ (blue polytope) on the sub-spaces. From the figure it is possible to observe that all vertexes are completely contained in the projections of the invariant ellipsoid $\mathcal{E}_\mathcal{C}$.

Using the values $T_{UC} = 0.37$ s and $\epsilon = 0.01$ the algorithm of software rejuvenation has been tested on the nonlinear model of the quadrotor controlled by the LQR controller developed using the linearized model of the system. Several tests have been carried out simulating attacks that provide arbitrary control inputs. In all the cases the reset clock t_r and safety controller keep the state of the system from hitting the boundary of $\mathcal{E}_\mathcal{C}$.

Figure **??** shows the position behaviour of the quadrotor respect the invariant ellipsoids, and Fig. **??** shows the zoom

of the behaviour respect the several modes of the controller. Figure 5 shows the time-line for the scheduling of the software rejuvenation modes in a scenario with two attacks. In the first case the attacker turns off all the propellers, and in the second the case the attacker tries to drive the system to a different equilibrium point.

The results show the effectiveness the adopted solution using $T_{UC} = 0.37$ s and $\epsilon = 0.01$ Fig.**??**. In Fig.5the controller is operating in MC (orange), SR (blue), SC (green), and MC under attack (red). It is important to note that, in the case of normal operation during the MC mode, the quadrotor remains in $\mathcal{E}_\epsilon$ after SR, so SC is not activated. If the quadrotor is out of $\mathcal{E}_\epsilon$ after SR, SC is activated and this happens in particular after the two attacks. Note that the SC activation does not require the attack detection; the activation strategy is based only on verifying the state condition $x^T P x \leq 1$.

VI. DISCUSSION

This paper presents conditions for safety controller design and timing parameters to support the implementation of software rejuvenation to provide CPS security against cyber attacks on the run-time code and data. The approach for the case of linear dynamics at an equilibrium state is developed the general results are illustrated using simulation of a quadrotor, demonstrating how safety can guaranteed without having to detect the cyber attacks. The results in this paper provide the foundation for implementing tracking control, where a supervisory generates the reference signal to the mission control algorithm in a manner that provides a sequence of equilibrium states for safety control. In contrast to the approach developed in [12], the safety controller executes only when it is needed, with an upper bound on the time required to return the system to a safe state, and the reachability set computations to determine the software refresh time are performed offline rather than at run time.

There are several directions for future research. The safety control time bound and the reset time can be improved using less conservative reachability algorithms. We are currently extending the results to incorporate state estimation, modeling uncertainties and disturbances. We are also developing experimental implementations and investigating methods for managing the equilibrium information and control limits on protected hardware components.

ACKNOWLEDGMENTS

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.
Carnegie Mellon® is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.
DM18-1192

APPENDIX

The state space representation of the dynamics of a continuous-time LTI system is

$$\dot{x}(t) = Ax(t) + Bu(t), \tag{6}$$

where $x \in \mathbb{R}^n$ is the state, and $u \in \mathbb{R}^p$ is the input.
A linear control that stabilizes (6) is

$$u(t) = -Kx(t) \tag{7}$$

where K is computed minimizing the cost function

$$J = \int_0^\infty x^T(\tau)Qx(\tau) + u^T(\tau)Ru(\tau)d\tau. \tag{8}$$

where $Q \geq 0$ and $R > 0$ are symmetric matrices. The minimum is achieved if

$$K = R^{-1}B^T S \tag{9}$$

where S is the solution of the associated Riccati algebraic equation [19]. Controller (7) is a linear quadratic regulator. The associated closed-loop system is

$$\dot{x}(t) = A_{SC}x(t) \tag{10}$$

where $A_{SC} \triangleq A - BK$.

Given a convex polyhedral region $\mathcal{C} \subset \mathbb{R}^n$ of the state space given by

$$\mathcal{C}(x_{eq}) = \left\{x \mid \xi_j^T(x - x_{eq}) \leq 1,\ j = 1, ..., n_c\right\} \tag{11}$$

the largest positively invariant ellipsoid respect to the equilibrium point x_{eq} and associated to (10)

$$\mathcal{E}_{P,SC}(x_{eq}) = \left\{x \mid \|(x - x_{eq})\|_P^2 \leq 1\right\} \tag{12}$$

contained in $\mathcal{C}$ can be found solving the following maximization problem for $x_{eq} = 0$ [20]:

$$\begin{cases} \max \log \det Q \\ s.t.\ QA_{SC}^T + A_{SC}Q \leq 0 \\ \quad \xi_j^T Q \xi_j \leq 1,\ j = 1, ..., n_c \\ \quad Q > 0 \end{cases} \tag{13}$$

where $P = Q^{-1}$ [20]. Set (12) is a positively invariant set, since

$$\|(x - x_{eq})\|_P^2 = (x - x_{eq})^T P (x - x_{eq}) \tag{14}$$

is a quadratic Lyapunov function. The set

$$\mathcal{E}_{\epsilon_{SC}}(x_{eq}) = \left\{x \mid \|(x - x_{eq})\|_P^2 \leq \epsilon\right\} \tag{15}$$

with $1 \leq \epsilon_{SC} \leq 1$ is a Lyapunov level-set where $\mathcal{E}_{\epsilon_{SC}}(x_{eq}) \subseteq \mathcal{E}_{P,SC}(x_{eq})$. Given (11) it is possible to find a convex polyhedron $\mathcal{C}_{\epsilon_{SC}}(x_{eq})\mathcal{E}_{\epsilon_{SC}}(x_{eq})$ as

$$\mathcal{C}_{\epsilon_{SC}}(x_{eq}) = \left\{x \mid \xi_j^T(x - x_{eq}) \leq \epsilon_{SC},\ j = 1, ..., n_c\right\}. \tag{16}$$

An approximation of the reachable set of (6) from the initial state $\mathcal{E}_{\epsilon_{SC}}(x_{eq})$, for all the possible controls $u \in \mathcal{U}$ at time t is

$$\mathcal{R}(\mathcal{C}_{\epsilon_{SC}}(x_{eq}), t; \mathcal{U}) = \{x | e^{At}x_0 + \textstyle\int_0^t e^{A(t-\tau)}Bu(\tau)d\tau, \\ x_0 \in \mathcal{C}_{\epsilon_{SC}}(x_{eq}),\ u(\tau) \in \mathcal{U},\ 0 \leq \tau \leq t\}.$$

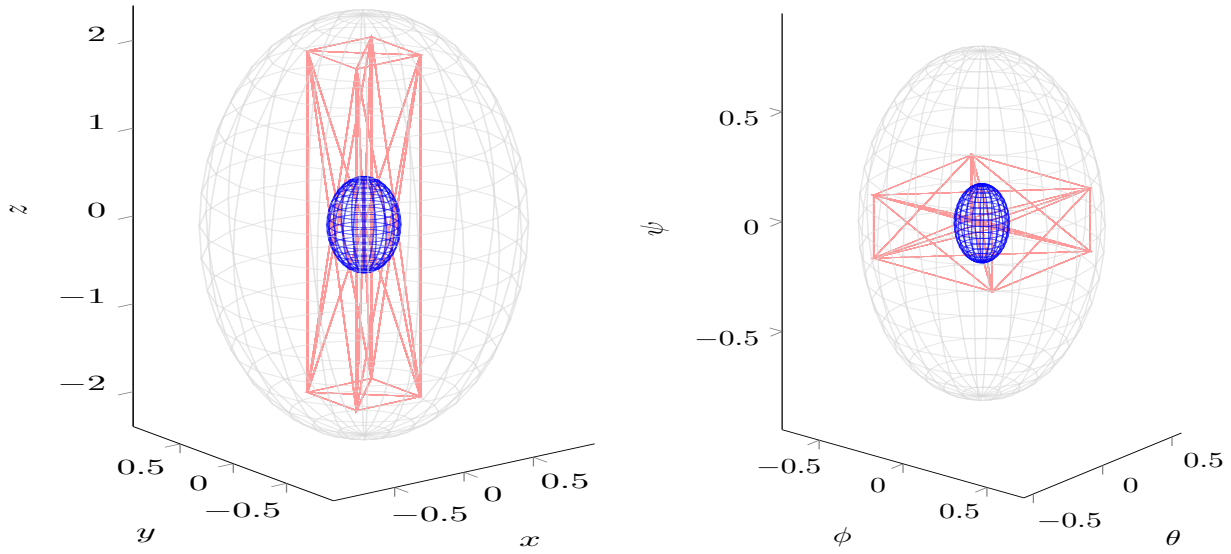

Fig. 3. Projections of $\mathcal{E}_{\mathcal{C}}$(gray), $\mathcal{E}_{\epsilon_{SC}}$ (blue) and $\mathcal{R}^{+}$ computed for $\epsilon_{SC} = 0.05$ and $T_{UC} = 0.34s$ (red line) on the position (left), and angles(right).

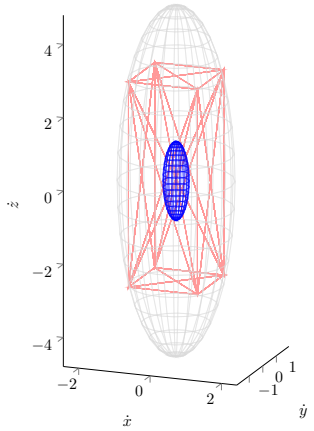

Fig. 4. Projections of $\mathcal{E}_{\mathcal{C}}$(gray), $\mathcal{E}_{\epsilon}$ (blue) and $\mathcal{R}^{+}$ computed for $\epsilon = 0.05$ and $T_{UC} = 0.34s$ (red line) on the linear velocity space.

An over-approximation of $\mathcal{R}(\mathcal{C}_{\epsilon_{SC}}(x_{eq})$ can be found using the method proposed in [21] for polytopic sets. The supporting hyperplanes to $\mathcal{R}(\mathcal{C}_{\epsilon_{SC}}(x_{eq})$ are

$$v_j^{+}(x,t) = \xi_j(t)x, \qquad j = 1, ..., n_c \tag{17}$$

where $\xi_j(0) = \xi_j$. Using the maximum principle of the optimal control [22], the normal direction of ξ_i is

$$\xi_j(t) = e^{-A^T\tau}\xi_j. \tag{18}$$

Representing $\mathcal{U}$ as convex combination of n_u vertexes $u_1, ..., u_{n_u}$, the over-approximation of $\mathcal{R}(\mathcal{C}_{\epsilon_{SC}}(x_{eq}), t, \mathcal{U})$ is [23]

$$\mathcal{R}^{+}(\mathcal{C}_{\epsilon_{SC}}(x_{eq}), t, \mathcal{U}) = \bigcap_{j=1}^{n_c} \Big\{ x | v_j^{+}(x,t) \leq \int_0^t \max_i \langle \xi_j(\tau), Bu_i \rangle d\tau + \max_{x(0) \in \mathcal{C}_{\epsilon_{\mathcal{SC}}}(\S_{\exists \amalg})} v_j^{+}(x(0), 0) \Big\} \tag{19}$$

The convex polyhedral set (19) can be also represented as polytope,

$$\mathcal{R}^{+}(\mathcal{C}_{\epsilon_{SC}}(x_{eq}), t, \mathcal{U}) = \left\{ x \in R^n \mid x = \sum_{k=1}^{n_v} \alpha_h x_k, \ \sum_{k=1}^{n_v} \alpha_k = 1 \ \alpha_k \geq 0 \right\} \tag{20}$$

References

[1] Alvaro A Cardenas, Saurabh Amin, and Shankar Sastry. Secure control: Towards survivable cyber-physical systems. In *28th International Conference on Distributed Computing Systems Workshops, 2008. ICDCS'08.*, pages 495–500. IEEE, 2008.

[2] Jill Slay and Michael Miller. Lessons learned from the maroochy water breach. In *International Conference on Critical Infrastructure Protection*, pages 73–82. Springer, 2007.

[3] Defense Use Case. Analysis of the cyber attack on the ukrainian power grid. *Electricity Information Sharing and Analysis Center (E-ISAC)*, 2016.

[4] Ralph Langner. Stuxnet: Dissecting a cyberwarfare weapon. *IEEE Security & Privacy*, 9(3):49–51, 2011.

[5] Y. Huang, C. Kintala, N. Kolettis, and N.D. Fulton. Software rejuvenation: Analysis, module and applications. In *Proceedings of 25th International Symposium on Fault Tolerant Computing*, pages 381–390, Pasadena, CA, June 1995. IEEE Computer Society.

[6] Miguel A. Arroyo, Lakshminarasimhan Sethumadhavan, and Jonathan Weisz. Secured cyber-physical systems, June 2017. United States Patent Number 15/618019 (pending).

[7] Miguel Arroyo, Hidenori Kobayashi, Simha Sethumadhavan, and Junfeng Yang. FIRED: frequent inertial resets with diversification for emerging commodity cyber-physical systems. *arXiv preprint arXiv:1702.06595*, 2017.

[8] Domenico Cotroneo, Roberto Natella, Roberto Pietrantuono, and Stefano Russo. A survey of software aging and rejuvenation studies. *J. Emerg. Technol. Comput. Syst.*, 10(1):8:1–8:34, January 2014.

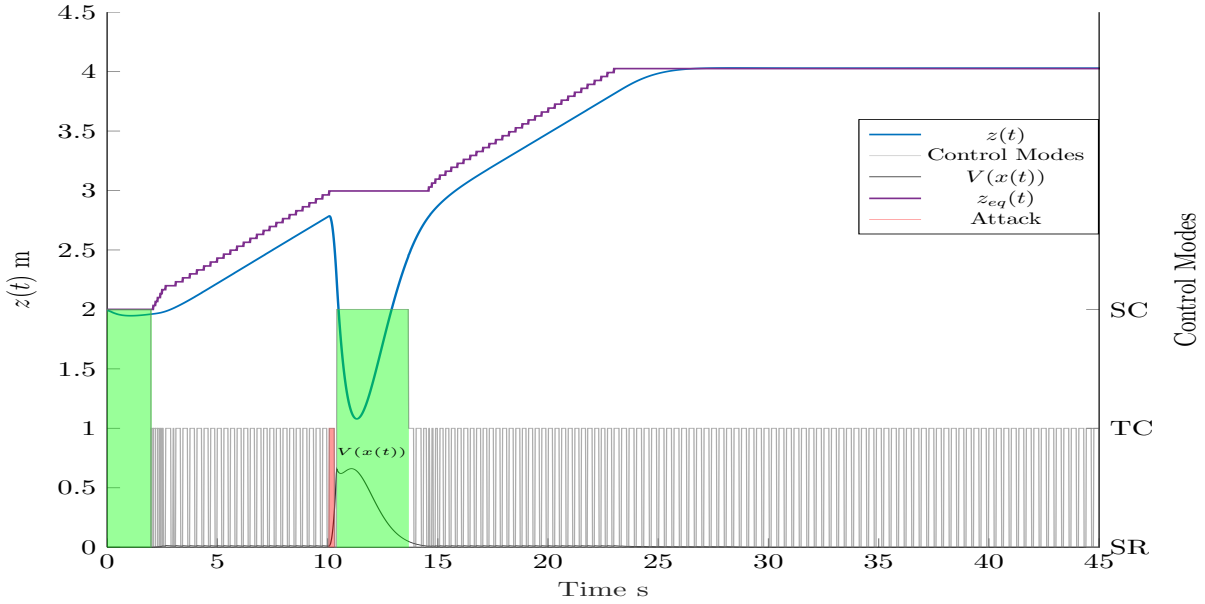

Fig. 5. System response along z respect to the time, the equilibrium points, the switching between the several control modes and the Lyapunov function $V(x(t))$.

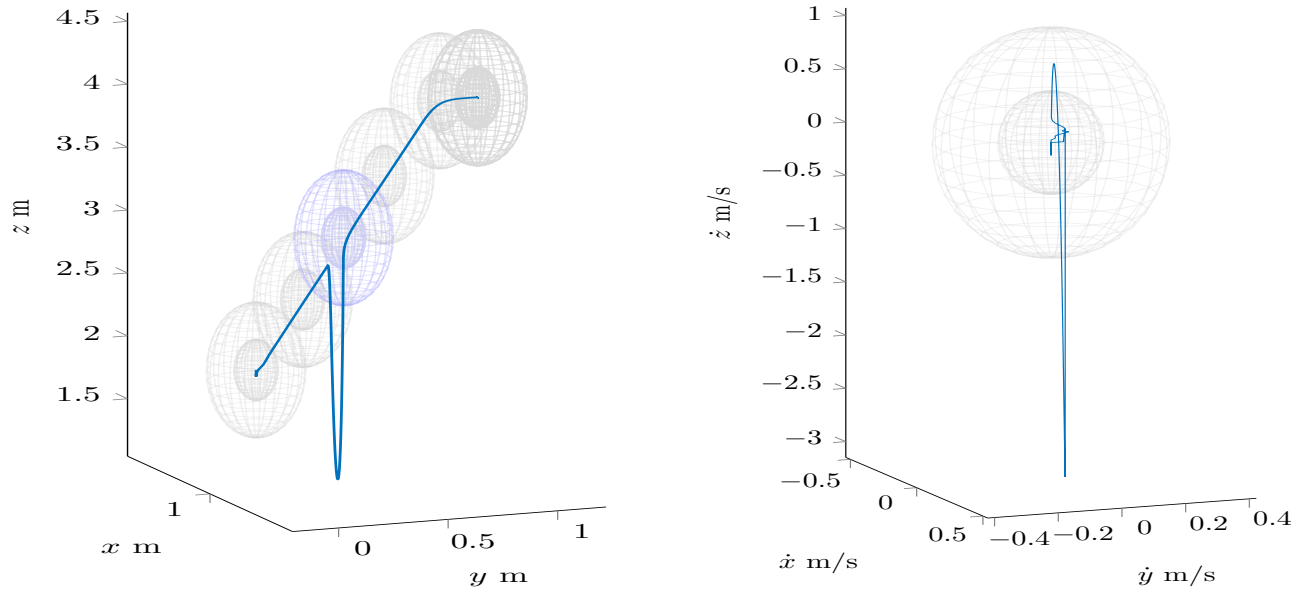

Fig. 6. Projections of $\mathcal{E}_\epsilon$(outer) and $\mathcal{E}_{\epsilon'}$ (inner) through the several updated equilibrium points on the position and linear velocity spaces. The blue line represents the respective system response.

[9] J. Alonso, A. Bovenzi, J. Li, Y. Wang, S. Russo, and K. Trivedi. Software rejuvenation: Do it amp;amp;amp; telco industries use it? In *2012 IEEE 23rd International Symposium on Software Reliability Engineering Workshops*, pages 299–304, Nov 2012.

[10] Khin Mi Mi Aung and Jong Sou Park. Software rejuvenation approach to security engineering. In Antonio Laganá, Marina L. Gavrilova, Vipin Kumar, Youngsong Mun, C. J. Kenneth Tan, and Osvaldo Gervasi, editors, *Computational Science and Its Applications – ICCSA 2004*, pages 574–583, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[11] Chen-Yu Lee, Krishna M. Kavi, Mahadevan Gomathisankaran, and Patrick Kamongi. Security through software rejuvenation. In *The Ninth International Conference on Software Engineering Advances ICSEA*, Nice, France, Oct 2014.

[12] Fardin Abdi, Chien-Ying Chen, Monowar Hasan, Songran Liu, Sibin Mohan, and Marco Caccamo. Guaranteed physical security with restart-based design for cyber-physical systems. In *Proceedings of the 9th ACM/IEEE International Conference on Cyber-Physical Systems*, ICCPS '18, pages 10–21, Piscataway, NJ, USA, 2018. IEEE Press.

[13] Jean-Piere Aubin. *Viability Theory*. Birkhäuser, 2009.

[14] Px4 flight controller: Generic 10" quad + geometry. `https://github.com/PX4/jMAVSim/blob/83bf400d71588131e2c6e179a6c63e8585271275/src/me/drton/jmavsim/Simulator.java#L314`, Last view: 09/2018.

[15] Randal Beard. Quadrotor dynamics and control rev 0.1. 2008.

[16] Oualid Araar and Nabil Aouf. Full linear control of a quadrotor uav, lq vs h. In *Control (CONTROL), 2014 UKACC International Conference on*, pages 133–138. IEEE, 2014.

[17] Hossein Bolandi, Mohammad Rezaei, Reza Mohsenipour, Hossein

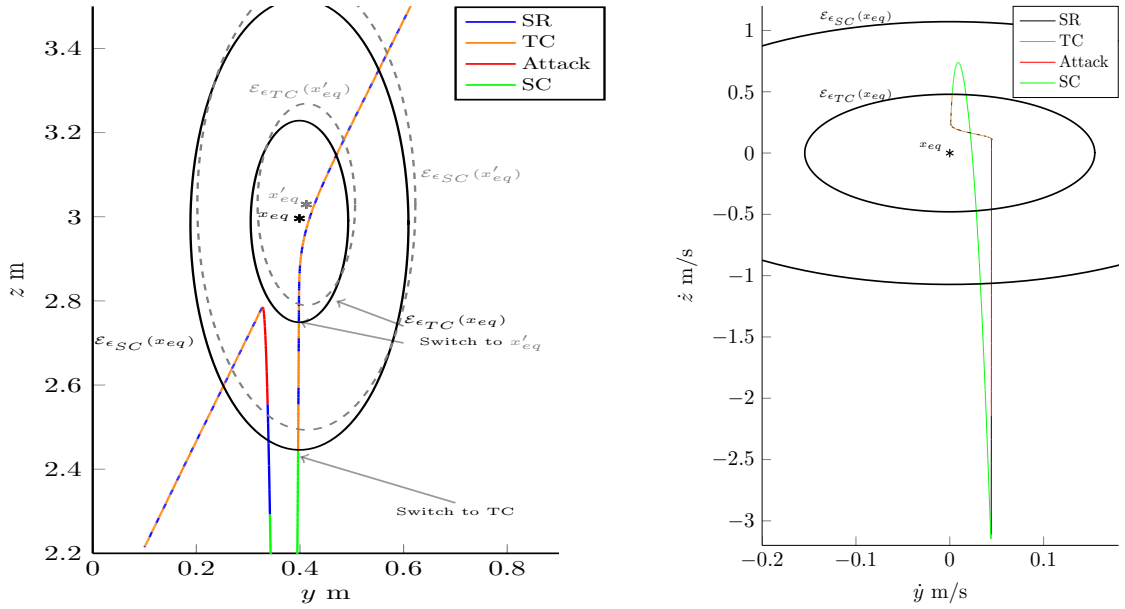

Fig. 7. The switching modes behaviour in correspondence of the attack projected on the $y-z$ and $\dot{y}-\dot{z}$ planes. On the right it is illustrated the switch between two consecutive equilibrium points.

$$P = \begin{bmatrix} 0.4185 & 0.1988 & 0 & 0 & 0 & 0 & 0 & 0 & -0.0712 & -0.6927 & 0 & 0 \\ 0.1988 & 1.1293 & 0 & 0 & 0 & 0 & 0 & 0 & -0.0448 & -0.5882 & 0 & 0 \\ 0 & 0 & 0.4185 & 0.1988 & 0 & 0 & 0.0712 & 0.6927 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.1988 & 1.1293 & 0 & 0 & 0.0448 & 0.5882 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.0435 & 0.0248 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.0248 & 0.1741 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.0712 & 0.0448 & 0 & 0 & 0.0588 & 0.2315 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.6927 & 0.5882 & 0 & 0 & 0.2315 & 3.0969 & 0 & 0 & 0 & 0 \\ -0.0712 & -0.0448 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0588 & 0.2315 & 0 & 0 \\ -0.6927 & -0.5882 & 0 & 0 & 0 & 0 & 0 & 0 & 0.2315 & 3.0969 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0400 & 0.0052 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0052 & 1.6218 \end{bmatrix} \tag{21}$$

$$K = \begin{bmatrix} 0 & 0 & 0 & 0 & -1.2773 & -0.7071 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.0313 & 0.7071 & 0 & 0 & 0.7342 & 3.91 & 0 & 0 & 0 & 0 \\ -1.0313 & -0.7071 & 0 & 0 & 0 & 0 & 0 & 0 & 0.7342 & 3.91 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.7161 & 0.7071 \end{bmatrix} \tag{22}$$

Nemati, and Seed Majid Smailzadeh. Attitude control of a quadrotor with optimized pid controller. *Intelligent Control and Automation*, 4(03):335, 2013.

[18] Samir Bouabdallah, Andre Noth, and Roland Siegwart. Pid vs lq control techniques applied to an indoor micro quadrotor. In *Proc. of The IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 2451–2456. IEEE, 2004.

[19] Frank L Lewis, Draguna Vrabie, and Vassilis L Syrmos. *Optimal control*. John Wiley & Sons, 2012.

[20] Stephen Boyd, Laurent El Ghaoui, Eric Feron, and Venkataramanan Balakrishnan. *Linear matrix inequalities in system and control theory*, volume 15. Siam, 1994.

[21] Inseok Hwang, Dušan M Stipanović, and Claire J Tomlin. Polytopic approximations of reachable sets applied to linear dynamic games and a class of nonlinear systems. In *Advances in control, communication networks, and transportation systems*, pages 3–19. Springer, 2005.

[22] Pravin Varaiya. Reach set computation using optimal control. In *Verification of Digital and Hybrid Systems*, pages 323–331. Springer, 2000.

[23] Alexander B Kurzhanski and Pravin Varaiya. Dynamic optimization for reachability problems. *Journal of Optimization Theory and Applications*, 108(2):227–251, 2001.